

Claims

1. A binary search tree comprising:

5 a multiplicity of address nodes each operable to store a data element, the nodes having pre-determined addresses and organised in a multiplicity of levels, the nodes including a root node and for each node at each level except the lowest level two child nodes in the immediately lower level, whereby the address of each child node is computable from the address of the respective node having that child node; and

10 a hardware engine for the insertion of elements in the nodes, said hardware engine being operable to make a search for the highest available node for the insertion of a new element and to search in a pattern in which all the nodes at each level beginning at the highest are searched before the search continues to the next lower level.

15 2. A binary search tree according to claim 1 wherein for each current node in said search the search comprises:

20 (a) determining whether a non-zero element is stored at the current node;

(b) determining, in the event that said non-zero element is stored, whether the current node is the last node at a current level of the tree;

25 (c) determining, if said current node is said last node, whether the current level is the lowest level of the tree;

(d) decrementing, if said current level is not the lowest level, the current level of the search and changing the current node to the first node of the next lower level of the tree; and

30 (e) setting, if said current node is not the last node at the current level, the current node to be the next node at the same level.

3. A binary search tree according to claim 2 wherein said engine, when the current node is available for the storage of a new element, causes the writing of a new element.

4. A binary search tree according to claim 2 wherein said engine, when said current node is available for the storage of a new element, is operative:

(i) to insert the new element if the current node is the root node, or when the current element is not the root node:

(ii) to determine whether the new element is greater or less than the element stored at the parent node of the current node and to increment or decrement the current node respectively; and

(iii) to insert the new element in accordance with an examination of the availability of the current node and a comparison of the magnitudes of the new element and the current element if any stored at the current node.

5. A method of establishing entries in a binary search tree, said binary search tree comprising a multiplicity of address nodes each operable to store a data element, the nodes having pre-determined addresses and organised in a multiplicity of levels, the nodes including a root node and for each node at each level except the lowest level two child nodes in the immediately lower level, whereby the address of each child node is computable from the address of the respective node having that child node:

said method comprising examining the nodes in a predetermined pattern to find a highest available node, said pattern requiring all the nodes at each level beginning at the highest to be examined before any node in the next lower level is examined.

6. A method according to claim 5 wherein said method comprises, for each current node that is examined:

(a) determining whether a non-zero element is stored at the current node:

5

()

(e) setting, if said current node is not the last node at the current level, the current node to be the next node at the same level.

12717	12718	12719	12720	12721	12722	12723	12724	12725	12726	12727	12728	12729	12730	12731	12732	12733	12734	12735	12736	12737	12738	12739	12740	12741	12742	12743	12744	12745	12746	12747	12748	12749	12750	12751	12752	12753	12754	12755	12756	12757	12758	12759	12760	12761	12762	12763	12764	12765	12766	12767	12768	12769	12770	12771	12772	12773	12774	12775	12776	12777	12778	12779	12780	12781	12782	12783	12784	12785	12786	12787	12788	12789	12790	12791	12792	12793	12794	12795	12796	12797	12798	12799	12800	12801	12802	12803	12804	12805	12806	12807	12808	12809	12810	12811	12812	12813	12814	12815	12816	12817	12818	12819	12820	12821	12822	12823	12824	12825	12826	12827	12828	12829	12830	12831	12832	12833	12834	12835	12836	12837	12838	12839	12840	12841	12842	12843	12844	12845	12846	12847	12848	12849	12850	12851	12852	12853	12854	12855	12856	12857	12858	12859	12860	12861	12862	12863	12864	12865	12866	12867	12868	12869	12870	12871	12872	12873	12874	12875	12876	12877	12878	12879	12880	12881	12882	12883	12884	12885	12886	12887	12888	12889	12890	12891	12892	12893	12894	12895	12896	12897	12898	12899	12900	12901	12902	12903	12904	12905	12906	12907	12908	12909	12910	12911	12912	12913	12914	12915	12916	12917	12918	12919	12920	12921	12922	12923	12924	12925	12926	12927	12928	12929	12930	12931	12932	12933	12934	12935	12936	12937	12938	12939	12940	12941	12942	12943	12944	12945	12946	12947	12948	12949	12950	12951	12952	12953	12954	12955	12956	12957	12958	12959	12960	12961	12962	12963	12964	12965	12966	12967	12968	12969	12970	12971	12972	12973	12974	12975	12976	12977	12978	12979	12980	12981	12982	12983	12984	12985	12986	12987	12988	12989	12990	12991	12992	12993	12994	12995	12996	12997	12998	12999	13000
-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------